The Relational Model 2

- Functional dependencies
    - Closure for attributes
    - Inference rules
- BCNF

Reading: Sections 3.1, 3.2, 3.3.1—3.3.3 of textbook.

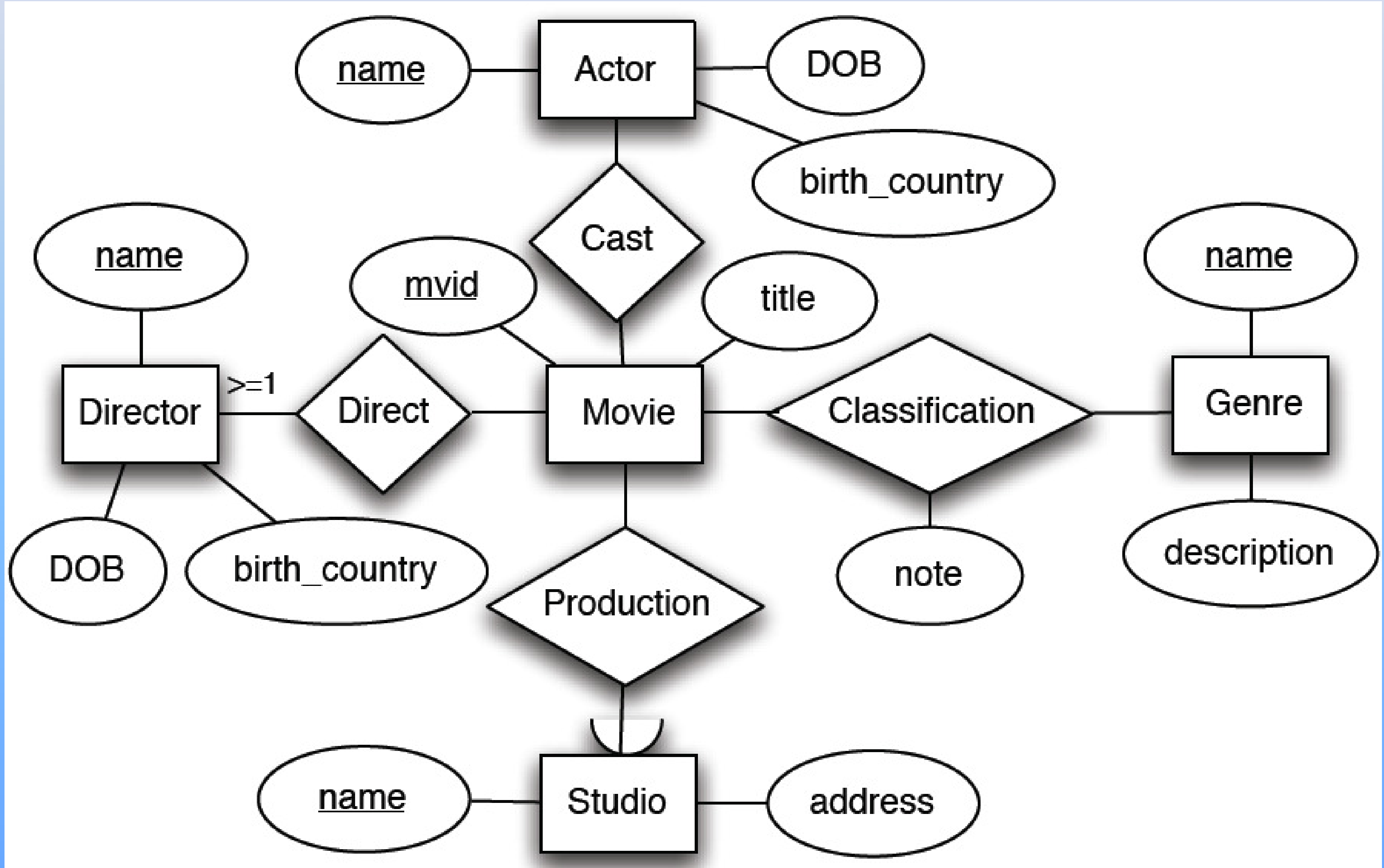
Relational Database Design

- Designing a relational database is about designing the schemata for relations --- how to organize attributes (information) into relations.
  - Design an ER diagram and then convert it into a relational database schema.
  - Design relation schemas directly from requirements.

Relational Database Design

- Is a database schema a good design?
- How good is it?
- How is the goodness of a database schema examined?
- How can be a bad design be improved?
  - Functional Dependency.
  - Normal forms.

The Village Cinema Database – ER diagram



The Village Cinema Database -- Relational Database Schema

Is the following database schema mapped from the ER diagram a good design?

Movie(mvID, Title)
Director(name, DOB, birth-country)
Actor(name, DOB, birth-country)
Genre(name, description)
Studio(name, address)
Classification(mvID*, Genre-name*, note)
Cast(mvID*, Actor-name*)
Direct(mvID*, Director-name*)
Production(mvID*, studio-name*)

Functional Dependency

- Functional dependency generalises the concept of keys of a relation:
  - A key of a relation is an assertion: when any two tuples of a relation agree on the values for attributes in a key, the two tuples agree with each other.
  - A key determines a relation (schema).
  - A functional dependency on a relation is an assertion: when any two tuples of a relation agree on a set of attributes, they agree on another set of attributes.

Functional Dependency ...

Functional dependency (FD)

X→*Y* (*X* determines *Y*)

is an assertion about a relation *R* that whenever two tuples of *R* agree on all the attributes of *X*, then they must also agree on all attributes in set *Y*. We say

X -> *Y* holds in *R*.

- *X* and *Y* represent sets of attributes.
- Attribute list represents a set without the brackets { and }.

Functional Dependency ...

- X->$A_1A_2 \ldots A_n$ holds for R exactly when each of X->A_1, X->A_2,…, X->A_n hold for R.
  - Note that in X->$A_1A_2 \ldots A_n$, $A_1A_2 \ldots A_n$ indeed represents {A_1, A_2, …, A_n}.
- X->$A_1A_2 \ldots A_n$ is equivalent to
  $X \rightarrow A_1$, $X \rightarrow A_2$, …, $X \rightarrow A_n$.
- Example:
  - A->BC can be split into A->B and A->C.
  - A → B and A→ C can be combined into A → B, C.

Example: FD

Given

Class(cno, title, day, time, room, type)

- FD 1:cno → title

Whenever two tuples agree on the values for Course, they must agree on the values for Title.

- FD 2: day, time, room → cno, type

which is equivalent to

day, time, room → cno

day, time, room → type

Example: Sample data

cno → title

day, time, room → type

cno	title	day	time	room	type
isys1057	Database Concepts	Wed	1030	14.04.27	lect
isys1057	Database Concepts	Thur	1330	14.04.27	lect
isys1055	Database Concepts	Wed	1730	12.05.02	lect
isys1057	Database Concepts	Wed	1130	14.10.30	tute
isys1057	Database Concepts	Wed	1330	14.09.23	tute
acct1009	Another Course	Wed	1130	14.04.27	lect
acct1009	Another Course	Thur	1330	07.02.23	lect
acct1009	Another Course	Thur	1430	07.02.23	tute

day, time, room → cno

Trivial FDs

- FDs that are always true (on whatever relations) are trivial FDs.
- An FD *X* → *Y* is a trivial FD if *X* includes *Y* (*Y* is a subset of *X*).
- Example:
  - course → course
  - course, title → course
  - day, time, room → day, time

Where do FDs come from?

- FDs are natural constraints on the relationship among information from real-world situation.
  - FDs do not depend on any sample data.
  - FDs are basic constraints at the attribute level. FDs constrain how data is related no matter how attributes are grouped into relations.
- Example: from the incomplete sample data for the Class relation below it may seem

  time → cno ✗

  but it is not true.

cno	title	day	time	room	type
isys1057	Database Concepts	Wed	1030	14.04.27	lect
isys1057	Database Concepts	Thur	1330	14.04.27	lect
isys1055	Database Concepts	Wed	1730	12.05.02	lect
isys1057	Database Concepts	Wed	1130	14.10.30	tute
isys1057	Database Concepts	Wed	1330	14.09.23	tute

The closure of an Attribute set

- Given a set of attributes $\{A_1, ..., A_n\}$ and a set of FDs S, the closure of $\{A_1, ..., A_n\}^+$ under S is the set of attributes B such that $A_1, ..., A_n \rightarrow B$ can be inferred from the FDs in S.

Finding the Closure for a set of Attributes

- Input: A set of attributes $\{A_1, ..., A_n\}$ and a set of FD's S.
- Output: the closure $\{A_1, ..., A_n\}^+$.

1. Split the FD's in S into FD's with single-attribute right hand side.
2. Initialise X to {A1, ..., An}.
3. Search for $B_1, ..., B_m \rightarrow C$ such that $\{B_1, ..., B_m\} \subseteq X$ but not C. Add C to X.
4. Repeat Step 3 until no more attributes can be added to X. Output X.

Example: the closure for a set of attributes

- Given

Class(cno, title, day, time, room, type}

  - day, time, room → cno, type
  - cno → title

- What is {day, time, room}$^+$?
  - {day, time, room, cno, title, type} – all attributes in relation Class.
- What is {cno}$^+$?
  - {cno, title}

Exercise: The Village Cinema Database Schema

For each relation below,

- What are the likely FDs?
- What is What is $\{mvID\}^+$?

Movie(mvID, Title)

Classification(mvID, Genre-name, note)

Production(mvID, studio-name)

Inferring FD's

- New FDs can be inferred from existing FDs.
- For example, given

    day, time, room → cno

    cno → title

It can be inferred that

    day, time, room → title

This rule is called the transitive rule, or transitivity. This rule is intuitively true. (See Section 3.2.6 of the text for a formal proof)

Inferring FDs – all rules

- Reflexivity:
  - Given sets of attributes X and Y, if $Y \subseteq X$, then $X \rightarrow Y$.
- Augmentation:
  - Given attributres, $A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_k$
    if $A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$, then
    $A_1, \ldots, A_n, C_1, \ldots, C_k \rightarrow B_1, \ldots, B_m, C_1, \ldots, C_k$
- Transitivity:
  - Given sets of attributes X, Y, and Z, if $X \rightarrow Y$, $Y \rightarrow Z$, then $X \rightarrow Z$.

Superkey and Key

- A superkey of a relation is any subset of attributes that uniquely determines the relation.
  - A relation can have several superkeys.
  - At least all attributes of the relation form a superkey for the relation.
- A key for a relation if it is a minimum superkey -- no proper subset of a key is a superkey.
  - A relation can several candidate keys, of which one is specified as the primary key.

Superkey and Key Revisited

Given a set of FDs on relation R,

K is a superkey for relation *R* if

$K \rightarrow R$, (or equivalently $K^+ = R$)

- Note that *R* is the set of all attributes in the relation.
- $K \rightarrow R$ if and only if $K \rightarrow R - K$.

Note that $R - K$ are all attributes of *R* not in *K*.

Superkey

Class(cno, title, day, time, room, type)

- cno → title
- Day, time, room → type, cno

- {cno, title, day, time, room} is a superkey because together these attributes determine all the other attributes.
  - cno, title, day, time, room → type
  - $\{day, time, room\}^+ = \{cno, title, day, time, room, type\}$
- Other superkeys:
  - {day, time, room}
  - {cno, day, time, room}
  - {cno, day, time, room, type}
  - Any other superset of {day, time, room}, including itself.
- The following are NOT superkeys:
  - {cno, day, time}
  - {cno, time, room, type}
  - {title, type, day}

Key

Class(cno, title, day, time, room, type)

- cno → title
- day, time, room → type, cno

- For any superkey, at least {day, time, room} must be included. {day, time, room} is a minimum superkey, because none of its subsets is a superkey.

  day, room → cno ✗

  day, time → room ✗

  time, room → type ✗

  {day, time, room} is a key for relation Class.
- There are no other keys for Class.

Relational Schema Design

- Goal of relational schema design is to remove redundancy and anomalies.
  - *Update anomaly* : one occurrence of a fact is changed, but not all occurrences.
  - *Deletion anomaly* : valid fact is lost when a tuple is deleted.

Bad Design leads to data redundancy

Class(cno, title, day, time, room, type)

cno	title	day	time	room	type
isys1057	Database Concepts	Wed	1030	14.04.27	lect
isys1057	Database Concepts	Thur	1330	14.04.27	lect
isys1055	Database Concepts	Wed	1730	12.05.02	lect
isys1057	Database Concepts	Wed	1130	14.10.30	tute
isys1057	Database Concepts	Wed	1330	14.09.23	tute
acct1009	Another Course	Wed	1130	14.04.27	lect
acct1009	Another Course	Thur	1330	07.02.23	lect
acct1009	Another Course	Thur	1430	07.02.23	tute

The sample data has data redundancy.

Bad Design Leads to Anomalies

cno	title	day	time	room	type
isys1057	Database Concepts	Wed	1030	14.04.27	lect
isys1057	Database Concepts	Thur	1330	14.04.27	lect
isys1055	Database Concepts	Wed	1730	12.05.02	lect
isys1057	Database Concepts	Wed	1130	14.10.30	tute
isys1057	Database Concepts	Wed	1330	14.09.23	tute
acct1009	Another Course	Wed	1130	14.04.27	lect
acct1009	Another Course	Thur	1330	07.02.23	lect
acct1009	Another Course	Thur	1430	07.02.23	tute

- Update anomaly: if the course isys1057 is renamed to "Database Fundamentals" during program renewal, this change must be made in all 4 places. Otherwise inconsistency occurs.
- Deletion anomaly: If acct1009 is deleted from the timetable this semester, the fact that there is a course acct1009 is lost.

Solution

- Measure the goodness of a relational schema:
  - A relaion in Boyce-Codd Normal Form (BCNF) is free from redundancies or anomalies.
- Normalisation: decompose an ill-designed relation into a set of relations in BCNF.

Boyce-Codd Normal Form

- We say a relation *R* is in BCNF if whenever *X* ->*Y* is a nontrivial FD that holds in *R*, *X* is a superkey.
  - Remember: *nontrivial* means *Y* is not contained in *X*.
  - Remember also that a *superkey.* is any superset of a key (not necessarily a proper superset). A key is special (minimal) superkey.

Example

Class(cno, title, day, time, room, type)

cno → title

day, time, room → cno, type

- Only key is {day, time room}.
- cno → title violates BCNF, as {cno} is not a superkey of Class.
- day, time, room → cno, type

  does not violate BCNF, as {day, time, room} is a superkey of Class.
- Class is not in BCNF.

Example …

Class is normalised (decomposed) into the following two relations (process to be explained later):

CourseInfo(cno, title)

CourseClass(day, time, room, cno*, type)

Note that the relations after decomposition are suitably named to reflect meaning of the relation.

- CourseInfo is in BCNF. The only projected FD:

  cno → title

  conforms BCNF. {cno} is the only key.
- CourseClass is in BCNF. The only projected FD:

  day, time, room → cno, type

  conforms BCNF. {day, time, room} is the only key.

Example …

Class

cno	title	day	time	room	type
isys1057	Database Concepts	Wed	1030	14.04.27	lect
isys1057	Database Concepts	Thur	1330	14.04.27	lect
isys1055	Database Concepts	Wed	1730	12.05.02	lect
isys1057	Database Concepts	Wed	1130	14.10.30	tute
isys1057	Database Concepts	Wed	1330	14.09.23	tute
acct1009	Another Course	Wed	1130	14.04.27	lect
acct1009	Another Course	Thur	1330	07.02.23	lect
acct1009	Another Course	Thur	1430	07.02.23	tute

Tuples projected to {cno, title}.

Tuples projected to {cno, day, time, room, type}.

ClassInfo

cno	title
isys1057	Database Concepts
isys1055	Database Concepts
acct1009	Another Course

CourseClass

cno	day	time	room	type
isys1057	Wed	1030	14.04.27	lect
isys1057	Thur	1330	14.04.27	lect
isys1055	Wed	1730	12.05.02	lect
isys1057	Wed	1130	14.10.30	tute
isys1057	Wed	1330	14.09.23	tute
acct1009	Wed	1130	14.04.27	lect
acct1009	Thur	1330	07.02.23	lect
acct1009	Thur	1430	07.02.23	tute

After decomposition, redundancy and anomalies are removed.

Exercise

Class(cno, title, day, time, room, type, staff)

- cno → title
- day, time, room → cno
- room → type (lecutre theatres and tute/lab rooms are for different purposes)

- What are the superkeys for Class?
- What are the keys for Class?
- Is Class in BCNF?

Exercise: The Village Cinema Database Schema

- Discuss likely FDs for each relation.
- Use FDs to examine the correctness of key and foreign key annotations for each relation.
- Is the database schema a good design? In other words, is the database in BCNF?

Movie(<u>mvID</u>, Title)
Director(<u>name</u>, DOB, birth-country)
Actor(<u>name</u>, DOB, birth-country)
Genre(<u>name</u>, description)
Studio(<u>name</u>, address)
Classification(<u>mvID*, Genre-name*</u>, note)
Cast(<u>mvID*, Actor-name*</u>)
Direct(<u>mvID*, Director-name*</u>)
Production(<u>mvID*</u>, studio-name*)